

# t8code - modular adaptive mesh refinement in the exascale era

Johannes Holke<sup>1¶</sup>, Johannes Markert<sup>1\*</sup>, David Knapp<sup>1\*</sup>, Lukas Dreyer<sup>1\*</sup>, Sandro Elweijer<sup>1\*</sup>, Niklas Böing<sup>1\*</sup>, Chiara Hergl<sup>1\*</sup>, Prasanna Ponnusamy<sup>1\*</sup>, Jakob Fussbroich<sup>1\*</sup>, Tabea Leistikow<sup>1\*</sup>, Florian Becker<sup>1\*</sup>, Ioannis Lilikakis<sup>1\*</sup>, Julia Weber<sup>1\*</sup>, Sven Goldberg<sup>1\*</sup>, Veli Ünlü<sup>1\*</sup>, and Achim Basermann<sup>1</sup>

<sup>1</sup> German Aerospace Center (DLR), Institute for Software Technology, Cologne, Germany ¶  
Corresponding author \* These authors contributed equally.

DOI: [10.xxxxxx/draft](https://doi.org/10.xxxxxx/draft)

## Software

- [Review](#)
- [Repository](#)
- [Archive](#)

Editor: [Open Journals](#)

## Reviewers:

- [@openjournals](#)

Submitted: 01 January 1970

Published: unpublished

## License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](#)).

## Summary

In this paper, we present our scalable dynamic adaptive mesh refinement (AMR) library t8code, which was officially released in 2022 (Holke et al., 2022). t8code is written in C/C++, open source, and readily available at [www.dlr-amr.github.io/t8code](https://www.dlr-amr.github.io/t8code). It is developed and maintained at the Institute for Software Technology of the German Aerospace Center (DLR). The software library provides fast and memory efficient parallel algorithms for dynamic AMR to handle tasks such as mesh adaptation, load-balancing, ghost computation, feature search and more. t8code can manage meshes with over one trillion mesh elements (Holke et al., 2021) and scales up to one million parallel processes (Holke, 2018). It is intended to be used as mesh management backend in scientific and engineering simulation codes paving the way towards high-performance applications of the upcoming exascale era.

## Statement of Need

Adaptive Mesh Refinement has been established as a successful approach for scientific and engineering simulations over the past decades (Babuvška & Rheinboldt, 1978; Bangerth et al., 2007; Dörfler, 1996; Teunissen & Keppens, 2019). By modifying the mesh resolution locally according to problem specific indicators, the computational power is efficiently concentrated where needed and the overall memory usage is reduced by orders of magnitude. However, managing adaptive meshes and associated data is a very challenging task, especially for parallel codes. Implementing fast and scalable AMR routines generally leads to a large development overhead motivating the need for external mesh management libraries like t8code.

Currently, t8code's AMR routines support a wide range of element types: vertices, lines, quadrilaterals, triangles, hexahedra, tetrahedra, prisms, and pyramids. The latter having a 1 : 10 refinement rule with tetrahedra emerging as child elements (Knapp, 2020). Additionally, implementation of other refinement patterns and element shapes is possible according to the specific requirements of the application. t8code aims to provide a comprehensive mesh management framework for a wide range of use cases in science and engineering applications.

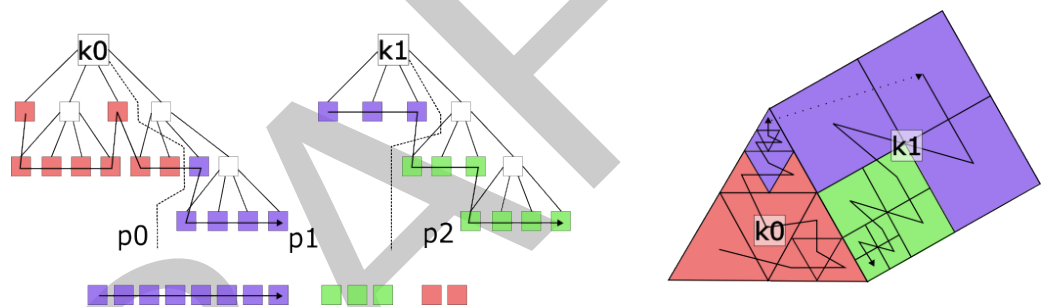
## Fundamental Concepts

t8code is based on the concept of tree-based adaptive mesh refinement. Starting point for the usage of t8code is an unstructured input mesh, which we denote a coarse mesh. This coarse

39 mesh describes the geometry of the computational domain. The coarse mesh elements are  
40 refined recursively in a structured pattern, resulting in refinement trees of which we store only  
41 minimal information of the finest elements (the leaf nodes of the tree). We call this resulting fine  
42 mesh the forest.

43 By enumerating the children in the refinement pattern we obtain a space-filling curve (SFC)  
44 logic. Via these SFCs, all elements in a refinement tree are assigned an index and are stored in  
45 linear order of these indices. Information such as coordinates or element neighbors do not need  
46 to be stored explicitly, but can be recovered from the index and the appropriate information of  
47 the coarse elements. The forest mesh can be distributed, that is, at any time, each parallel  
48 process only stores a unique portion of the forest mesh, the boundaries of which are calculated  
49 from the SFC indices; see Figure 1.

50 While in the past being successfully applied to quadrilateral and hexahedral meshes (Burstedde  
51 et al., 2011; Weinzierl, 2019), t8code extends these SFC techniques in a modular fashion,  
52 such that arbitrary element shapes are supported. We achieve this modularity through a  
53 novel decoupling approach that separates high-level (mesh global) algorithms from low-level  
54 (element local) implementations. All high-level algorithms can then be applied to different  
55 implementations of element shapes and refinement patterns. A mix of different element shapes  
56 in the same mesh is also supported.



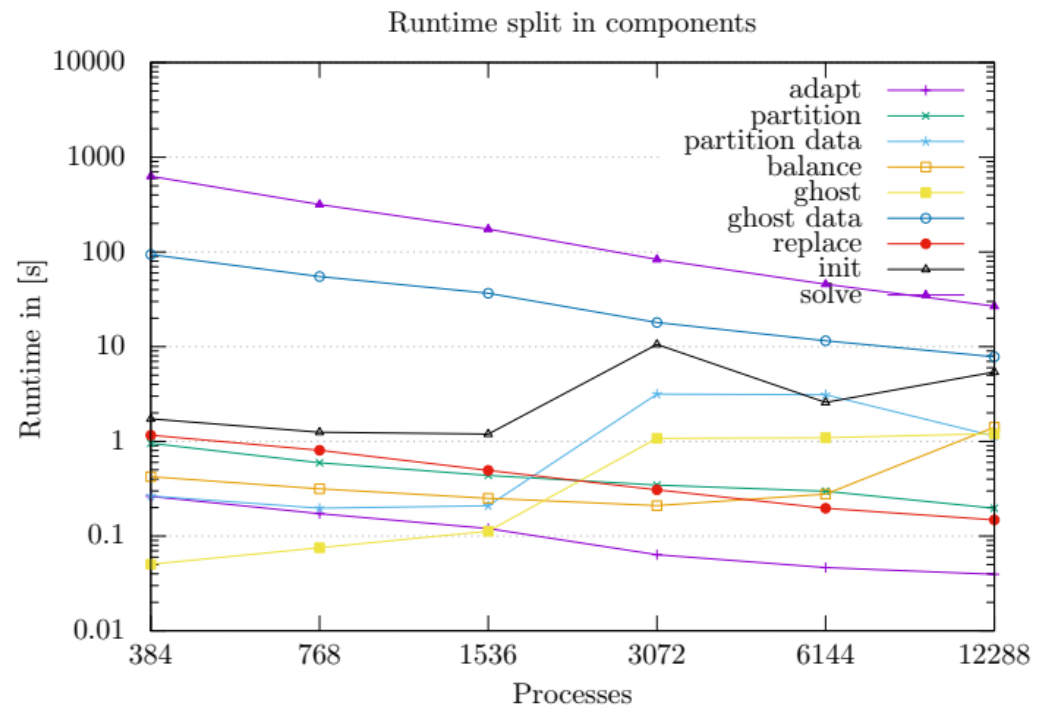
**Figure 1:** Left: Quad-tree of an exemplary forest mesh consisting of two trees ( $k_0$ ,  $k_1$ ) distributed over three parallel processes  $p_0$  to  $p_2$ . The SFC is represented by a black curve tracing only the finest elements (leaf nodes) of each tree. Right: Sketch of the associated mixed shape mesh refined up to level three.

## 57 Performance

58 t8code supports distributed coarse meshes of arbitrary size and complexity, which we tested for  
59 up to 370 million input elements (Burstedde & Holke, 2017). Moreover, we present some of our  
60 benchmark results from various performance studies conducted on the JUQUEEN (JUQUEEN  
61 Supercomputer, n.d.) and the JUWELS (JUWELS Supercomputer, n.d.) supercomputers at  
62 the Jülich Supercomputing Center. t8code's ghost and partition routines are exceptionally  
63 fast with proper scaling of up to 1.1 trillion mesh elements; see Table 1, (Holke et al., 2021).  
64 Furthermore, in a prototype code (Dreyer, 2021) implementing a high-order discontinuous  
65 Galerkin method (DG) for advection-diffusion equations on dynamically adaptive hexahedral  
66 meshes we observe a speed-up of 12 compared to non-AMR meshes with only an overall 15%  
67 runtime contribution of t8code; see Figure 2.

# Process	# Elements	# Elem. / process	Ghost	Partition
49,152	1,099,511,627,776	22,369,621	2.08 s	0.73 s
98,304	1,099,511,627,776	11,184,811	1.43 s	0.33 s

Table 1: Runtimes on JUQUEEN for the ghost layer and partitioning operations for a distributed mesh consisting of 1.1 trillion elements.



**Figure 2:** Runtimes on JUQUEEN of the different components of our DG prototype code coupled with t8code. Note that all features associated with dynamical mesh adaptation utilize only around 15% of the total runtime largely independent of the number of processes.

## Conclusion

In this note, we introduce our open source AMR library t8code. We give a brief overview of the fundamental design principles and high-level operations. Due to the high modularity, t8code can be easily extended for a wide range of use cases. Performance results confirm that t8code is a solid choice for mesh management in high-performance applications in the upcoming exascale era.

Even though t8code is a newcomer to the market, it is already in use as the mesh management backend in various research projects, most notably in the earth system modeling (ESM) community. In the ADAPTEX project t8code is integrated with the Trixi framework (Schlottke-Lakemper et al., 2020) - a modern computational fluid dynamics code written in Julia. Over the next years several ESM applications are planned to couple to this combination, including MESSy, MPTrac, and SERGHEI. Moreover, t8code also plays an important role in several DLR funded projects, e.g., VISPLORE, HYTAZER, Greenstars and PADME-AM.

For further information beyond this short note and also for code examples, we refer to our Documentation and Wiki reachable via our homepage [www.dlr-amr.github.io/t8code](http://www.dlr-amr.github.io/t8code) and our technical publications on t8code (Becker, 2021; Burstedde & Holke, 2016, 2017; Dreyer, 2021; Elsweijer, 2021; Fußbroich, 2023; Holke, 2018; Holke et al., 2021, 2022; Knapp, 2020; Lilikakis, 2022).

## Acknowledgement

Development and maintenance of t8code is mainly funded by the Institute of Technology of the German Aerospace Center.

The authors state that there are no conflicts of interest.

## References

- 90
- 91 Babuvška, I., & Rheinboldt, W. C. (1978). Error estimates for adaptive finite element  
92 computations. *SIAM Journal on Numerical Analysis*, 15(4), 736–754. <https://doi.org/10.1137/0715049>  
93
- 94 Bangerth, W., Hartmann, R., & Kanschat, G. (2007). Deal.II—a general-purpose object-  
95 oriented finite element library. *ACM Trans. Math. Softw.*, 33(4), 24–es. <https://doi.org/10.1145/1268776.1268779>  
96
- 97 Becker, F. (2021). *Removing hanging faces from tree-based adaptive meshes for numerical*  
98 *simulations* [Master's thesis]. Universität zu Köln.
- 99 Burstedde, C., & Holke, J. (2016). A tetrahedral space-filling curve for nonconforming adaptive  
100 meshes. *SIAM Journal on Scientific Computing*, 38, C471–C503. <https://doi.org/10.1137/15M1040049>  
101
- 102 Burstedde, C., & Holke, J. (2017). Coarse Mesh Partitioning for Tree-Based AMR. *SIAM Jour-*  
103 *nal on Scientific Computing*, Vol. 39, C364–C392. <https://doi.org/10.1137/16M1103518>
- 104 Burstedde, C., Wilcox, L. C., & Ghattas, O. (2011). p4est: Scalable Algorithms for Parallel  
105 Adaptive Mesh Refinement on Forests of Octrees. *SIAM Journal on Scientific Computing*,  
106 33(3), 1103–1133. <https://doi.org/10.1137/100791634>
- 107 Dörfler, W. (1996). A convergent adaptive algorithm for poisson's equation. *SIAM Journal on*  
108 *Numerical Analysis*, 33(3), 1106–1124. <https://doi.org/10.1137/0733054>
- 109 Dreyer, L. (2021). *The local discontinuous galerkin method for the advection-diffusion*  
110 *equation on adaptive meshes* [Master's thesis, Rheinische Friedrich-Wilhelms-Universität  
111 Bonn]. <https://elib.dlr.de/143969/>
- 112 Elswijker, S. (2021). *Curved Domain Adaptive Mesh Refinement with Hexahedra*. Hochschule  
113 Bonn-Rhein-Sieg. <https://elib.dlr.de/143537/>
- 114 Fußbroich, J. (2023). *Towards high-order, hybrid adaptive mesh refinement: Implementa-*  
115 *tion and evaluation of curved unstructured mesh elements* [Master's thesis, Technische  
116 Hochschule Köln]. <https://elib.dlr.de/200442/>
- 117 Holke, J. (2018). *Scalable algorithms for parallel tree-based adaptive mesh refinement with*  
118 *general element types* [{PhD} thesis]. Rheinische Friedrich-Wilhelms-Universität Bonn.
- 119 Holke, J., Burstedde, C., Knapp, D., Dreyer, L., Elswijker, S., Uenlue, V., Markert, J., Lilikakis,  
120 I., & Boeing, N. (2022). *t8code* (Version 1.0.0). <https://doi.org/10.5281/zenodo.7034838>
- 121 Holke, J., Knapp, D., & Burstedde, C. (2021). An Optimized, Parallel Computation of the  
122 Ghost Layer for Adaptive Hybrid Forest Meshes. *SIAM Journal on Scientific Computing*,  
123 C359–C385. <https://doi.org/10.1137/20M1383033>
- 124 *JUQUEEN supercomputer*. (n.d.). FZ Jülich. Retrieved January 3, 2023, from [https://hbp-hpc-platform.fz-juelich.de/?page\\_id=34](https://hbp-hpc-platform.fz-juelich.de/?page_id=34)  
125
- 126 *JUWELS supercomputer*. (n.d.). FZ Jülich. Retrieved January 3, 2023, from <https://www.fz-juelich.de/en/ias/jsc/systems/supercomputers/juwels>  
127
- 128 Knapp, D. (2020). *A space-filling curve for pyramidal adaptive mesh refinement* [{M}aster's  
129 Thesis]. Rheinische Friedrich-Wilhelms-Universität Bonn.
- 130 Lilikakis, I. (2022). *Algorithms for tree-based adaptive meshes with incomplete trees* [Master's  
131 thesis, Universität zu Köln]. <https://elib.dlr.de/191968/>
- 132 Schlottke-Lakemper, M., Gassner, G. J., Ranocha, H., & Winters, A. R. (2020). *Trixi.jl:*  
133 *Adaptive high-order numerical simulations of hyperbolic PDEs in Julia*. <https://github.com/trixi-framework/Trixi.jl>. <https://doi.org/10.5281/zenodo.3996439>  
134

- 135 Teunissen, J., & Keppens, R. (2019). A geometric multigrid library for quadtree/octree  
136 AMR grids coupled to MPI-AMRVAC. *Computer Physics Communications*, 245, 106866.  
137 <https://doi.org/https://doi.org/10.1016/j.cpc.2019.106866>
- 138 Weinzierl, T. (2019). The Peano Software-Parallel, Automaton-based, Dynamically Adaptive  
139 Grid Traversals. *ACM Transactions on Mathematical Software*, 45(2), 1–41. <https://doi.org/10.1145/3319797>  
140

DRAFT